VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2015

Ondřej Korbela

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ondřej Korbela**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: dSoft Solutions s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

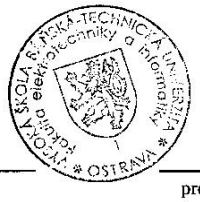
Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Pavla Dráždilová, Ph.D.**

Konzultant bakalářské práce: Gustáv Hlaváč

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2015

.....

Rád bych na tomto místě poděkoval především paní Mgr. Pavle Dráždilové, Ph.D., za její trpělivost a její ochotu se vším pomoci, protože bez ní by tato práce nevznikla.

Abstrakt

Tato bakalářská práce popisuje průběh mé praxe ve firmě Dsoft Solututions s.r.o.

Popisuje společnost, ve které jsem absolvoval odbornou praxi. Jsou zde uváděny použité technologie, se kterými jsem se setkal v průběhu praxe a také popsány prostředí, ve kterém jsem pracoval. V další kapitole se popisují jednotlivé technologie, se kterými jsem se setkal a které jsem využíval ve své práci. Tento popis je detailnější a podrobnější. Důraz je kladen na popis technologie, zejména kvůli tomu, že se jedná o nové technologie ve světě webových aplikací. V kapitole vlastní práce popisuji průběh této praxe, kde vysvětluji a popisuji zadané úlohy a jejich řešení. Na závěr hodnotím, jaký přínos pro mně měla praxe v dané firmě, jestli jsem se již s danou technologií seznámil a jestli zde jsem uplatnil znalosti ze školy.

Klíčová slova: Amazon web hosting, Elementary OS , WebStorm, MongoDB, Node.js

Abstract

This thesis deals with the course of my practice in the firm Dsoft Solututions Ltd.

I first describe the society in which I practiced and our first acquaintance. I mention the technology used and with whom I met during practice and also describe an environment in which I worked. The next chapter describes the technologies I have encountered and which I used in my work. This description is more detailed. Emphasis is placed on the description of the technology, especially because it is a new technology in the world of web applications. In the chapter own work I describe my way through this practice, which explains and describes assignments and their solutions. In conclusion, I evaluate what kind of benefit for me was the practice in the company, if I have already met with the technology and if I applied the knowledge from school.

Keywords: Amazon web hosting, Elementary OS , WebStorm, MongoDB, Node.js

Seznam použitých zkratek a symbolů

HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheets
EC2	–	Amazon Elastic Compute Cloud
JSON	–	JavaScript Object Notation
API	–	Application Programming Interface
HTTP	–	Hypertext Transfer Protocol
ORM	–	Object-relational mapping
SQL	–	Structured Query Language
CRUD	–	Create, Read, Update, Delete
DOM	–	Document Object Model
NTFS	–	New Technology File System
CGI	–	Computer-generated imagery
URI	–	Uniform Resource Identifier

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Odborná praxe byla vykonávaná ve firmě Dsoft Solutions s.r.o.

V průběhu této praxe jsme dostávaly úkoly, které se týkaly jak programování za pomocí NodeJS v aplikaci WebStorm a vytváření databáze v NoSQL MongoDB, tak i nastavování Linux serverů na Cloudu EC2 od Amazonu. Vyzkoušely jsme si nové a velice zajímavé technologie, jako např. Polymer-Project. Základním úkolem bylo naučit se dané technologii porozumět a pochopit jak a kde ji využit. S některými technologiemi jsme byli již předem seznámeni, jiné nám byly velkou neznámou. Úkoly jsme dostávali přiřazeny postupně podle nastudovaných technologiích, abychom si ověřily své znalosti. Na začátku této práce je popsána společnost, ve které bych odbornou praxi vykonával a popis činnosti. V úvodní části je popsáno navázání kontaktu a první střetnutí. Poté jsou popsány použité technologie, které bylo nutné si nastudovat a jejich využití v projektech. Následuje popis vlastní práce, která byla vykonána v průběhu praxe.

Dále je také v závěru zhodnoceno, jestli znalosti ze školy byly dostačující nebo naopak u daných technologií znalosti chyběly. Následně je popsáno, jestli odborná praxe byla přínosem a zdali práce byla přínosem pro společnost.

2 O firmě a jejím zaměření

Obrázek 1: Logo firmy

Dsoft Solutions s.r.o. je společnost, která se zabývá vývojem webových a mobilních aplikací. Nedílnou součástí je správa serverů. Tato společnost se zabývá informačními technologiemi od roku 2003. Od roku 2008 je také členem Švýcarské asociace pro kvalitu neboli SAQ. Pro tuto společnost není důležitá kvantita, ale kvalita. Snaží se vždy, aby se software přizpůsobil v maximální možné míře zákazníkům a jejím potřebám. Hlavní je zaměření na moderní trendy webových aplikací, mobilních aplikací a správě serverů.

2.1 Seznámení se s firmou

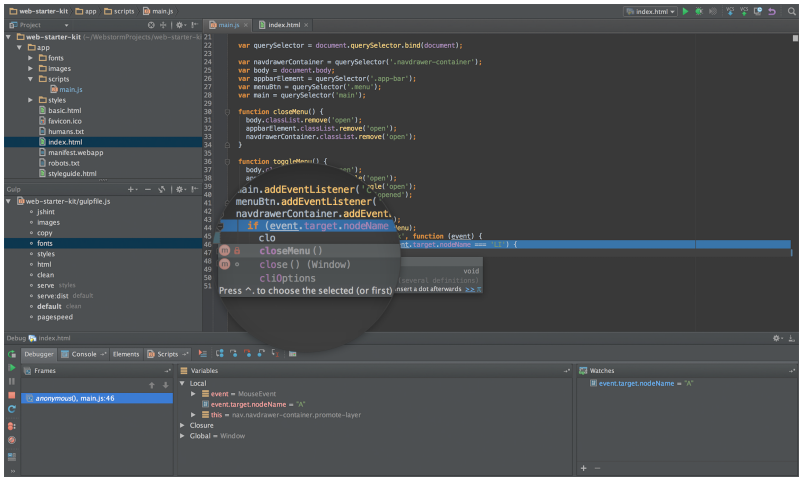
Jako první následovala video konference na Google plus, protože společnost nesídlí v České Republice. První bylo probráno zaměření společnosti, v jakých oblastech pracuje. Následně nám byl na sdílen soubor, ve kterém byly popsány technologie, se kterýma se v průběhu praxe seznámíme. U každé technologie bylo políčko, do kterého jsme museli vepsat status, jestli jsme se s danou technologií někdy setkali nebo je to pro nás novinka a tedy je potřeba vše nastudovat od úplného začátku. Následně jsme se bavili o operačním systému, který by bylo vhodné používat. Shodli jsme se na tom, že nejlepší je OSX na Macu, ale kvůli ceně není dostupný a tak jsme se domluvili, že budeme používat operační systém Linux. Jako poslední částí tohoto video hovoru byla domluva o používané aplikaci pro programování. Firma používá aplikaci WebStorm a tak jsme se tedy domluvili, že budeme tuto aplikaci využívat. Následně jsme obdrželi licenční klíč od aplikace WebStorm k jejímu nainstalování.

2.2 Využité technologie

Na Google plus jsme obdrželi dokument, se kterými technologiemi se budeme potýkat a které bychom si měli nastudovat. Začali jsme studiem příkazů v operačním systému Linux, kde většina práce spočívá v konzoli, to znamená psaní příkazů. Jako další jsme si nastudovali Git a vytvořil GitHub účet. Následně jsme si museli nakonfigurovat aplikaci WebStorm k využití NodeJS a AngularJS.

2.3 WebStorm

Program WebStorm, i když se to na první pohled nezdá, pochází z dílny firmy JetBrains, která začínala v roce 2001 v České Republice. WebStorm je komerční vývojové prostředí pro JavaScript, CSS a HTML. Od vydání páté verze je zde možnost takzvaného Live Edit,



Obrázek 2: Ukázka práce ve WebStorm

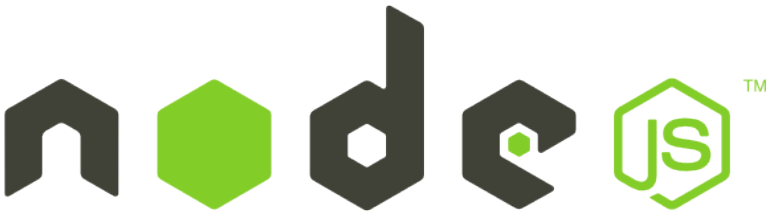
díky kterému vidíme editovanou stránku v reálném čase. V nadcházející verzi číslo šest byla aplikace nadále vylepšena a získala podporu editování HTML5. Tato aplikace je určena zejména pro práci s projekty. Je zde k dispozici řada framworků, např. Boostrap, Boilerplate pro HTML5 a další. Od verze šest nám umožňuje práci s nejmodernějšími jazyky, kódovat můžeme v CoffeeScript, TypeScript s automatickým převodem na JavaScript. Samozřejmě podporuje transformace LESS a Sass do CSS.

3 Použité technologie

3.1 Git a Github

Git je distribuované uložiště s verzováním. Funguje na principu repozitářů. Repozitář slouží pro uložení všech dat z projektu. Výhodou je to, že můžeme verzovat jednotlivé verze softwaru. Kvůli strachu vývojařů z velkého objemu dat, které by museli mít u sebe, vznikla speciální datová uložiště pro vývojáře, která umožňují skladování dat na serverech poskytovatele. Díky tomu vznikl GitHub, který nabízí neomezený prostor pro projekty s otevřenou licencí. Avšak pro hostování projektů k soukromému užití je potřeba zaplatit malý poplatek. Jako první nám bylo řečeno založit si GitHub účet. Poté jsme měli za úkol nastudovat git versioning solution. Základem bylo stažení balíčku, a jeho následná instalace. Poté následovalo prostudování jednotlivých příkazů, které pracují s Gitem. Výpis několika příkazů: git config –global user.name "[name]"- nám nastaví jméno ke commit transakcím git init [project-name] - vytvoří nám nový lokální repozitář se specifickým jménem git clone [url] – stáhne celý projekt a její historii verzí git status – vypíše nám nové nebo modifikované soubory, které jsou ke commitnutí Nastudovali jsme si také videa na youtube, jak správně pracovat s gitem a jeho příkazy. Naistalovali jsme si Sublime text. Jedná se o textový editor pro kód.

3.2 Node.js



Obrázek 3: Logo NodeJS

Node.js (server-side in JavaScript) je vysoce výkonné, událostmi řízené prostředí pro JavaScript. Základem je javascriptový interpret z Google Chrome. Nad ním je vrstva kódu v C++ poskytující minimální nutné zázemí jako jsou vyhodnocující příchozí události, obsluha I/O bufferů a jiné. Uživatelské rozhraní pro Node je čistý JavaScript v jediném vlákně. Node.js tedy umožňuje používat JavaScript na serveru. Zajímavostí Node.js je komunita, kde v roce 2011 bylo k dispozici 1800 modulů, které řešily vše důležité, co tvůrce aplikace potřeboval a o rok později je jich k dispozici 15 tisíc. Jedním příkazem můžeme jakýkoliv z nich naistalovat a ihned používat. Rád bych také zmínil zá-

kladní případy užití, kde Noje.js najde své uplatnění. Node.js je velmi vhodné použít, pokud chcete vytvářet tyto typy projektů. První je aplikace s RESTful/JSON API. Druhý je single-page aplikace (např. ve spolupráci s vynikajícím frameworkem AngularJS). Za třetí real-time aplikace (Socket.io, frameworky Derby.js či Meteor). Naopak Node.js se nehodí pro projekty, které jsou velmi náročné na CPU. Protože Node.js pracuje pouze v jednom vlákně, při spuštění aplikace (před zpracováním prvního HTTP požadavku) dojde k načtení a inicializaci všeho potřebného a jakmile je vše načteno, Node.js poslouchá příchozí požadavky a ty pak směřuje na konkrétní kontroléry a akce (v případě MVC architektury). Npm (node package manager) je balíčkovací systém. V Node.js balíčky (moduly) instalujeme podobným způsobem jako např. programy v Linuxu přes nástroj apt. S npm se pracuje přes klasický příkazový řádek a všechny moduly se instalují jednoduchým příkazem npm install. Frameworky v Node.js lze rozdělit do tři skupin. První skupina jsou frameworky inspirované světem Ruby, sem například patří Express, který je inspirován Ruby frameworkem Sinatra, dále např. RailwayJS či TowerJS, které jsou inspirované RubyOnRails. Druhou skupinou jsou frameworky pro real-time aplikace např. Derby, SocketStream. Třetí skupinu tvoří ostatní frameworky např. Flatrion od Nodejitsu, Mojito od Yahoo či Locomotive. Framework Express je pravděpodobně nejpoužívanějším frameworkem v Node.js, který mimo jiné používá např. sociální síť Linkedln.

3.3 HTML5 a CSS3

Obrázek 4: Logo HTML5 a CSS3

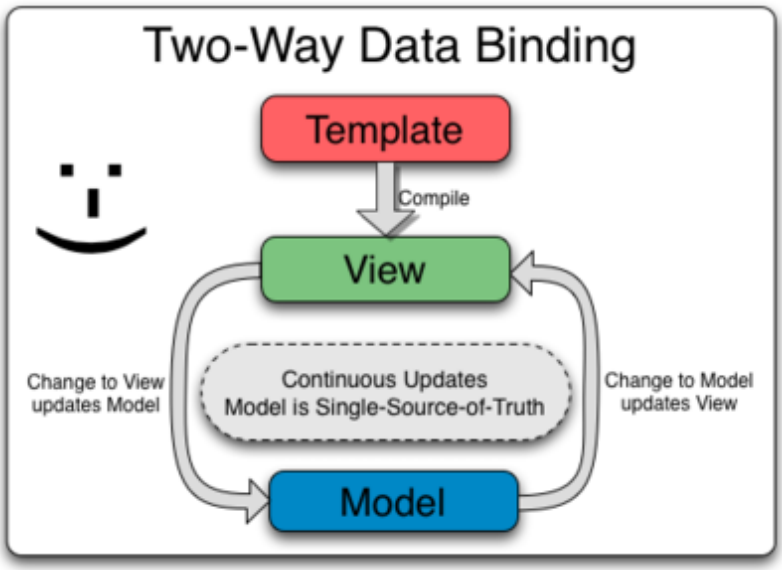
Technologie HTML5 A CSS3 jsou nezbytnou součástí všech webových stránek. HTML je jazyk pro rozvržení a propojení dokumentů, který definuje syntaxi a umístění prvků a také definuje hyperlinkové propojení dokumentů. Je navrženo pro strukturování dokumentu a obsahuje také prostředky pro členění dokumentu. Říká, jak a co se má zobrazovat. Pro stylování se používá CSS (Cascading Style Sheets) neboli tabulky kaskádových stylů. Definují vzhled celé skupiny elementů. Tento jazyk slouží pro popis způsobu zob-

razení stránek napsaných v jazycích HTML, XHTML nebo XML. Hlavním smyslem je oddělit vzhled dokumentu od jeho struktury a obsahu. Výhody, které přináší CSS jsou, že nabízí rozsáhlejší možnosti formátování, jednotný styl, oddělení struktury a stylu, dynamickou práci se stylem, formátování XML dokumentů, kompatibilita webových prohlížečů a kratší doba načítání stránky. Zpátky k HTML5, specifikace HTML5 je složena z několika nezávislých částí, např.: nové HTML značky sémanticky definující strukturu stránky a podpora off-line aplikací. Liší se od svého předchůdce HTML4 hlavně novými zkrácenými a rychlejšími zápisy značek. Hlavní změna je nová specifikace typu dokumentu, tedy DOCTYPE. U CSS3 je dneska již hodně využívaná nová vlastnost zaoblení rohů u HTML prvků. Díky této vlastnosti se zaoblení nemusí řešit pomocí obrázků, pouze musíme uvést parametry pro všechny prohlížeče, které chceme, aby byly podporovány. Tyto vlastnosti nazýváme animace. Další nový prvek je transformace, která umožňuje transformovat prvek. Rozlišují se tři transformace a to translate, rotate a scale. Translate umožňuje pohyb prvku po ose x a y. Rotate umožňuje rotaci prvku okolo své osy ve směru hodinových ručiček. Scale umožňuje změnu velikosti prvku. CSS3 přináší také nové vlastnosti jako nové modely RGBA, HSL a průhlednost.

3.4 AngularJS

Obrázek 5: Logo AngularJS

Další dílčí technologií je Framework AngularJS (client-side in JavaScript). AngularJS obsahuje řadu zajímavých myšlenek, přičemž mezi nejužitečnější koncepty frameworku patří Two way data-binding, implementace Dependency injection, testovatelnost, direktivy a znovu použitelnost komponent. Two way Data-binding bychom mohli do čestiny přeložit jako dvoucestná synchronizace dat, který řeší synchronizaci stavů mezi modelem a view. AngularJS nabízí deklarativní způsob implementace. To znamená, že programátor nemusí popisovat jak data synchronizovat, namísto toho pouze deklaruje „vztah“ a AngularJS nám zajistí synchronizaci. Šablona nám říkám jak aplikace má vypadat, model je objekt reprezentující entitu a pohled vznikne, když se do šablony vloží data z modelu (transformace na DOM). Dependency Injection je návrhový vzor, který řeší závislost mezi jednotlivými komponentami programu. V AngularJS je zabudovaný systém, který řeší DI napříč celou vyvíjenou aplikací. Umožňuje vždy přesně specifikovat, které jiné komponenty jsou uvnitř konkrétní komponenty používané.

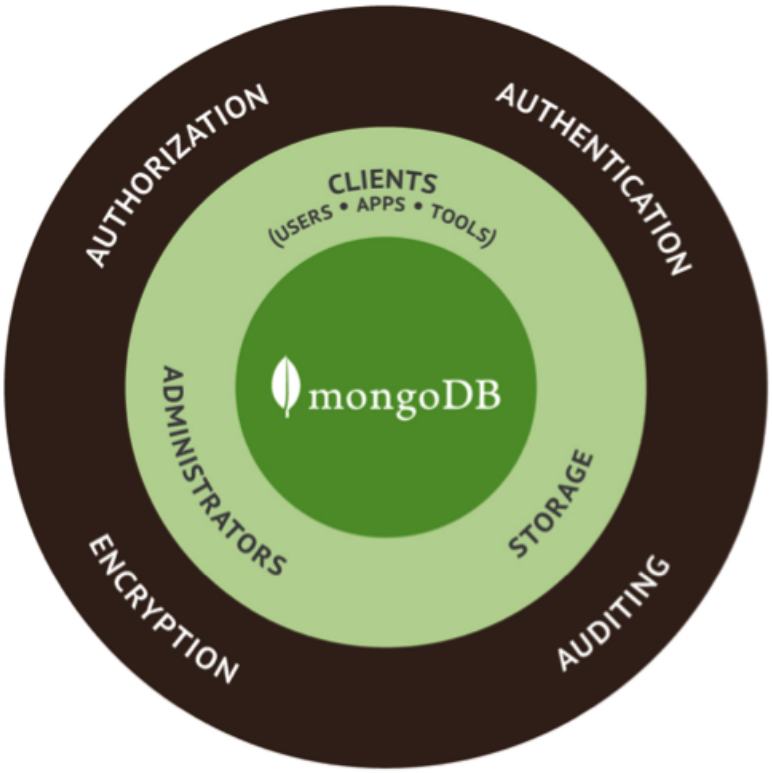


Obrázek 6: Obousměrné vázání dat v AngularJS

3.5 NoSQL MongoDB

MongoDB je jedna z nejpopulárnějších NoSQL databází. NoSQL je databázový koncept, ve kterém datové uložiště i zpracování dat používají jiné prostředky než tabulková schémata tradiční relační databáze. Motivací k tomuto přístupu mohou být jednoduchost designu, horizontální i vertikální škálovatelnost a jemnější kontrola dostupnosti. Databáze bez SQL jsou často vysoce optimalizovaná úložiště typu klíč-hodnota (ne vždy). Díky odlišné striktuře ukládání dat (např. stromová, grafová) oproti RDBMS, je i algoritmická složitost pro různé operace odlišná. Obecně se vhodnost aplikace daného typu databáze liší podle řešení problému. Segment NoSQL databází v současnosti roste a prospívá především v oblasti big data a real-time webu. A nyní zpět k MongoDB, který komunikuje přes JavaScript a data si databáze ukládá ve formátu BSON (Binary JSON). Začít pracovat s MongoDB je mnohem snazší, než začít s relační databází, kde je potřeba nejprve se naučit SQL. V MongoDB máme databáze, které obsahují kolekce, které dále obsahují libovolný počet dokumentů, kde dokument je v tomto případě objekt ve formátu JSON. Oproti SQL, které obsahují několik tabulek s jasně danou strukturou, které pak obsahují libovolný počet záznamů. V MongoDB není potřeba předem vytvářet žádnou strukturu kolekce a dokonce není potřeba ani vytvářet danou kolekci. Ta se vytvoří automaticky při prvním vložení dokumentu. Pokud chceme data z kolekce, která ještě v databázi neexistuje, vrátí se prázdný počet výsledků, nikoliv chyba. V mongoDB mohou být data libovolně zanořená. Např. kolekce s produkty může obsahovat dokument produkt, který bude mít několik potomků (sub dokumentů) variant. V MongoDB neexistuje spojování jako v SQL, vše je směřováno k tomu, aby se data dala získat jednoduchým dotazem. Tedy např. parametry produktů budou včetně názvů uloženy v kolekci s produkty. Do-

chází tedy k duplicitě dat. Budeme-li chtít změnit název parametru produktu, musíme dotazem projít všechny záznamy a parametr změnit.



Obrázek 7: Architektura zabezpečení MongoDB

3.6 Mongoose

Mongoose je Node.js modul, který umožňuje jednodušší přístup k objektům v MongoDB. Nabízí podobnou funkcionalitu jako různé ORM frameworky, např. Doctrine či ZendDB table pro PHP. Mongoose nabízí nástroje pro CRUD operace (vytváření, čtení, editaci a mazání) nebo třeba provádět validaci před vložením dokumentu do kolekce.

3.7 Polymer-Project

Web Components ohlašují novou éru vývoje webových aplikací založených na zapouzdření a interoperabilních vlastních prvcích, které rozšiřují samotné HTML. Polymer tedy usnadňuje a zrychluje vytvoření čehokoliv od tlačítka až po velké a ucelené aplikace. Už od začátku webu, měly prohlížeče svou vlastní výchozí sadu prvků. Většina z nich, stejně jako toho moc neudělaly. Ale některé prvky jsou docela silné např. <select>. Prohlížeč ví, co má dělat s tímto prvkem, když narazí na <select>, značkovací jazyk vytvoří

Obrázek 8: Logo Polymer-Project

interaktivní ovládání pro uživatele, které je opakovatelně použitelné. <Select> je balík prvků opakovaně použitelných, které navíc nemusíte provádět sami. Každá knihovna v JavaScriptu ví, jak komunikovat s DOM prvky. Můžeme si nastavit jeho chování s atributy HTML, bez použití skriptu. Pokud chytíme prvek z DOM, který má své metody a vlastnosti pro věci, které nedávají smysl značkovacímu jazyku. Nejenom, že můžeme zahrnout <select> uvnitř většiny ostatních druhů prvků, jeho chování se může měnit v závislosti na věci, které jsme do něho vložili. Prvky jsou základním stavebním kamenem webu. Bohužel, jak se webové aplikace stávají komplexnější, tak jsme přerostli základní sadu prvků, které se dodávají v prohlížečích. Řešení polymer-projektu je nahradit značky částí skriptu. V tomto posunu, ale ztratili eleganci prvků, avšak nesmíme to chápat, že se snaží zahrnout značky skripty, ale jako vybudování silnějších prvků. Sada nových výkonných technologií nazvaných Web Components toto umožňuje. Některé z vlastností prvků jsou UI-specifické, ale většina z nich není. Elementy mohou sloužit jako generický balík pro jedno použití funkčnosti. K dispozici máme tři koncepční vrstvy polymeru. První z nich je Web Components, na kterých je polymer postavený. Zatím ne všechny prohlížeče podporují tyto funkce, takže polyfilní vrstva se snaží vyplnit tyto mezery. Implementace rozhraní API je v jazyce JavaScript. Za běhu, polymer automaticky vybere nejrychlejší cestu - nativní implementace nebo JavaScript. Druhá z nich je knihovna polymer, která poskytuje deklarativní syntaxi, která nám to zjednoduší. Definuje vlastní prvky a přidává funkce, jako obousměrné vázání dat, sledování vlastností a podpora gest, které pomáhají vybudovat silné a opakovaně použitelné prvky. Poslední vrstvu tvoří elementy. Element tvoří komplexní sadu UI a non-prvků uživatelského rozhraní, které je možné hned používat. Tyto prvky jsou závislé na knihovně polymer, ale jsou samostatné a volitelné. Můžeme použít prvky bez použití přímo polymeru, nebo použít polymer k vytvoření vlastních prvků.

4 Vlastní práce

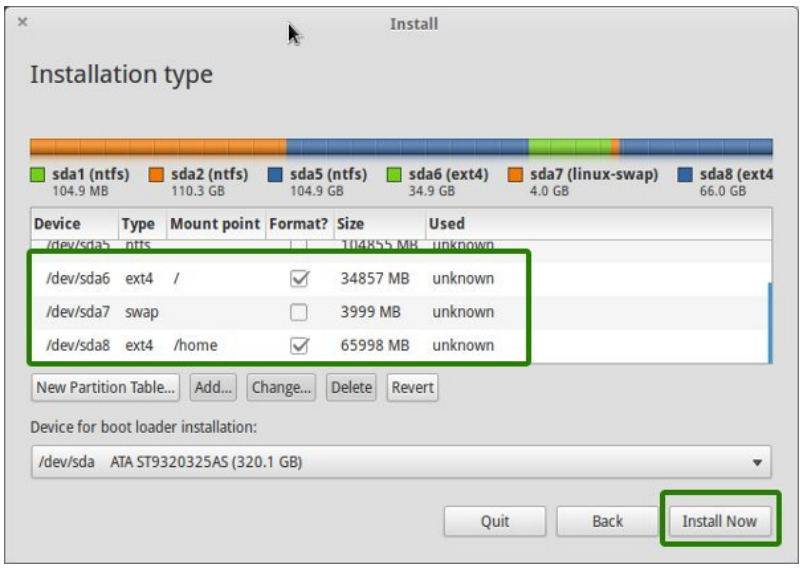
Jako první následoval videohovor přes platformu Google plus, kde jsme probrali technologie, které bychom si měli nastudovat a obecně probrali problematiku, kterou se budeme zabývat. Dále jsme probrali v jakém operačním prostředí je asi nejlépe pracovat, přičemž nejlépe z toho vyšel Mac, ale ten je docela drahý a tak jsme se dohodli, že budeme pracovat v prostředí s operačním systémem Linux. Já jsem začal používat Elementary OS. Domluvili jsme se, kterou aplikaci budeme využívat pro programování a poté jsme obdrželi licenční klíč k nainstalování aplikace WebStorm. V tomto prostředí jsme museli nakonfigurovat a nainstalovat potřebné technologie. Dále jsem si musel nakonfigurovat Amazon web services, kde jsme si museli nastavit EC2 teda virtuální server na cloudu.

4.1 Harmonogram

9.9. - 10.9.2014 Úvodní diskuze o průběhu praxe
16.9. - 17.9.2014 Videohovor na Google Plus
23.9 - 24.9.2014 Seznámení s firmou
30.9. - 31.9.2014 Seznámení se s distribucí Linuxu Elementary OS
6.10. - 7.10.2014 Instalace a konfigurace Elementary OS
13.10. - 14.10.2014 Seznámení se s aplikací WebStorm a její instalace
20.10. - 21.10.2014 Nastudování technologie Polymer-Project
27.10. - 28.10.2014 Práce s Polymer-Projectem
4.11. - 5.11.2014 Vytváření aplikace v Polymer-Projectu
11.11. - 12.11.2014 Pokračování ve vytváření aplikace
18.11. - 19.11.2014 Pokračování ve vytváření aplikace
25.11. - 26.11.2014 Dokončení aplikace v Polymer-Project
2.12. - 3.12.2014 Nastudování Node.js, Express a npm
9.12. - 10.12.2014 Aplikace Node.js
16.12. - 17.12.2014 Seznámení s Mongoose a MongoDB
5.1. - 6.1.2015 Instalace Mongoose a MongoDB
12.1. - 13.1.2015 Seznámení s Amazon web services
19.1. - 20.1.2015 Konfigurace Amazon web services
26.1. - 27.1.2015 Nastudování Git a GitHub
5.2. - 6.2.2015 Instalace a aplikace repozitáře Git a GitHub
12.2. - 13.2.2015 Vytvoření uživatele a konfigurace na serveru
19.2. - 20.2.2015 Start velkého projektu pro spediční společnost
26.2. - 27.2.2015 Vytváření databáze v MongoDB
2.3. - 3.3.2015 Pokračování v tvorbě databáze, tvoření objektů
9.3. - 10.3.2015 Tvorba databáze, vytvoření modelu
16.3. - 17.3.2015 Dokončení tvorby databáze

4.2 Instalace Elementary OS

Jako první úkol jsme si museli nainstalovat a nakonfigurovat operační systém Elementary OS, který je distribucí Linuxu a je založen na Ubuntu. Vybrali jsme si tuto distribuci, protože nám na pohled připomínala jablečný OS X a také zároveň vévodila jeho jednoduchost. Nainstalovali jsme si Elementary OS jako dual boot s windows, protože nám to dává možnost záchrany v případě, že bychom si něco pokazili na Elemetary OS. Jako první jsme si tedy museli stáhnout Elementary OS a vytvořit z něho live USB. Poté jsme restartovali počítač a klávesou F10 jsme se dostali to boot menu, kde jsme nastavili možnost boot z USB. Po chvilce strpení se nám podařilo nabootovat systém z USB. Na úvodní straně máme na výběr dvě možnosti a to vyzkoušet Elementary OS nebo nainstalovat Elementary OS. Na další obrazovce si vybereme jazyk a poté nám zkontroluje,



Obrázek 9: Elementary OS konečný vzhled oddílů

zdali máme aspoň 4,4 GB místa na disku, jestli jsme připojení k síti a jestli máme připojení k internetu. Když jsme se tímhle proklikali, přišla na řadu správa oddílu, tedy to nejdůležitější. Nejprve musíme smazat NTFS nebo existující ext4 oddíl a tím vytvořit volné místo. Zde vybere velikost rootu, vybereme ext4 souborový systém a nastavíme bod mountu na /root. Potom co jsme úspěšně vytvořili root, swap a home oddíly stačí už pouze nainstalovat.

4.3 Projekt pro spediční společnost

K tomu, abychom se mohli podílet na projektu, nám bylo řečeno, že jako první si musíme vytvořit AWS účet, se kterým jsme chvíli bojovali, ale nakonec se nám ho podařilo úspěšně založit. Bylo zde potřeba vložit i údaje o kreditní kartě, která musela být povolená. K tomu abychom měli virtuální server od Amazonu dostupný jsme si museli nakon-

figurovat a vytvořit instanci, která vyžadovala vytvoření klíče. Po úspěšném vytvoření instance jsme si vytvořili oddíl s velikostí 10 GB a přiřadili ho jíž vytvořené instanci. Poté jsme si museli chvíli počkat, než se vše nakonfiguruje a spustí, asi do půl hodiny již vše běželo, jak mělo a bylo připraveno k použití. Dále následovalo připojení k serveru, na kterém běžel operační systém Linux s distribucí Ubuntu. K tomu, abychom mohli komunikovat s tímto serverem na operačním systému Windows, jsme používali aplikaci Putty. Aplikaci jsme si museli nejprve nakonfigurovat. Jako hostname jsme museli uvést veřejnou IP adresu mého serveru a připojení bylo pomocí SSH. Ještě jednu důležitou věc bylo třeba nastavit a to sice svůj veřejný klíč, který jsme použili již dříve na konfiguraci serveru. V Putty je potřeba soubor s příponou .ppk a na serveru je potřeba soubor s příponou .pem. Použili jsme proto aplikaci přímo od Putty s názvem Puttygen, který nám byl schopen vygenerovat soubor s příponou .ppk. Následně jsme ho vložili do konfigurace Putty a mohlo se vše otestovat. Spojení bylo úspěšné a nám se podařilo dostat na server. Při připojení na server tedy dochází k autentizaci veřejným klíčem, kdy je ještě nutné zadat přístupové heslo. Poté můžeme začít s nastavováním.

Začali jsme instalací Nginx, Git a Gitwebu. Gitweb poskytuje vynikající rozhraní pro Git projekty, i když se využívá jenom lokálně. Poskytuje velmi pohodlný způsob, jak procházet repozitáře. První jsme si museli naistalovat Git pomocí příkazu sudo apt-get install git a k tomu jsme si nastaviti jméno a e-mail pomocí git config –global user.name "Ondřej Korbela"a podobně i e-mail jenom se změněným user.email. Následuje instalace webového serveru, na kterém tohle všechno poběží. Nainstalovali jsme si Nginx pomocí příkazu sudo apt-get install nginx. To nám naistalovalo Nginx server do /etc/nginx a také vytvořilo kontrolní skript v /etc/init.d/nginx. Server se nám bude zapínat defaultně, to si můžeme vyzkoušet, když najedeme na http://localhost, kde bychom měli vidět zprávu, která říká "Vítejte v Nginx!". Dále jsme si museli nainstalovat CGI (fast cgi wrapper) a také to co nám zařídí, aby se nám fcgi instance objevovaly tam kde mají. Toho jsme dosáhli zadáním příkazu sudo apt-get install fcgiwrap spawn-fcgi. Tímto jsme dosáhli toho, že se nám vytvořil socket soubor v /var/run/fcgiwrap.socket. Tento socket je unixovým socketem a zajišťuje komunikaci mezi nginx a fcgi procesy. .Následuje instalace gitwebu. Gitweb jsme si nainstalovali pomocí příkazu sudo apt-get install gitweb. Toto nám vytvořilo následující soubory a složky. V /etc/gitweb.conf máme konfiguraci Gitweb instance, jako defaultně je nastaven projekt z /var/cachce/git. V /usr/share/gitweb nám to vytvořilo složku, která obsahuje cgi skripty, obrázky a ikony. Jako poslední ve /var/cache/git máme defaultní pozici, na které Gitweb hledá projekty. Toto nastavení si můžeme změnit editací souboru v /etc/gitweb.conf. Poslední částí bylo vše napojit dohromady. Socket, který cgi wrapper vytvořil je vlastněn www-data, takže Nginx musí běžet jako tento uživatel. Abychom tohohle docílili, museli jsme v souboru /etc/nginx/nginx.conf změnit vlastníka na www-data. Nginx server potřebuje přijímat instrukce k servírování obsahu přes Gitweb.

```
server {
 listen  80 default;

location  /index.cgi  {
root  /usr/share/gitweb/;
```

```
include fastcgi_params;
gzip off;
fastcgi_param SCRIPT_NAME {\$}uri;
fastcgi_param GITWEB_CONFIG /etc/gitweb.conf;
fastcgi_pass unix:/var/run/fcgiwrap.socket;
}
location / {
root /usr/share/gitweb/;
index index.cgi;
}
}
```

Výpis 1: Konfigurace gitwebu

Finální částí je aktivování tohoto hostitele a poté restartování nginx.

```
sudo ln -s /etc/nginx/sites-available/gitweb /etc/nginx/sites-enabled/gitweb
sudo /etc/init.d/nginx restart
```

Výpis 2: Aktivování hostitele a restart nginx

Když se vše provedlo v pořádku, tak bychom měli na http://localhost vidět prázdný seznam projektů. Přidání projektů se provádí ve složce /var/cache/git. Naštěstí nemusíme mít své veškeré projekty v této složce, ale stačí, když nalinkujeme Git složku z projektu do této složky. Takže, když si budeme chtít přidat složku s názvem projects, musíme použít příkaz sudo ln -s /projects/.git /var/cache/git/projects.git. Poté stačilo pouze znovu načíst stránku s projekty a už ho máme v seznamu. Samozřejmě nechceme, aby nám byl každý schopen procházet projekty a hlavně jak to rozhraní Gitwebu povoluje. Když nechceme, aby nám někdo procházel projekty, můžeme nastavit přístupová práva Nginx. Nejjednodušší je přistup pomocí IP adresy.

```
allow 91.127.20.26;
allow 213.81.142.41;
allow 127.0.0.1;
deny all;
```

Výpis 3: Nastavení IP adresy

Tímto povolíme přístup localhostu a dalším dvěma IP adresám. Pro lepší správu jsme si vytvořili nového uživatele s názvem Git a k němu aplikovali práva, která umožnují práci s repozitáři. Dále musíme nastavit nutná práva k tomu, abychom se mohli dostat k repozitáři. Posléze je nutné ještě nakonfigurovat své repozitáře.

```
useradd -m git
su git
git clone --bare /old/repo.git /home/git/repositories/repo.git
cd /home/git/repositories/repo.git
sudo chmod -R g+ws .
sudo chgrp -R git .
git --bare update-server-info
cp hooks/post-update.sample hooks/post-update
chmod a+x hooks/post-update
```

Výpis 4: Nastavení repozitáře

Toto nám vygeneruje veškeré informace, které jsou potřebné ke sdílení repozitáře za pomocí webserveru jako je Nginx.

4.4 Práce s Polymer-Projectem

Práce s Polymer-Projectem je velmi zajímavá. Měli jsme za úkol vytvořit jednoduchou polymer aplikaci, tedy základního klienta pro službu sociálních sítí. Jako první jsme si museli stáhnout jejich základní balíček, který obsahuje začáteční verzi projektu polymer, se kterým budeme pracovat. K tomu abychom mohli pracovat, potřebujeme základní HTTP server. Nejlepší je použití python pomocí příkazu python -m http.server. Nyní bychom měli mít přístup k našemu projektu, což si můžeme vyzkoušet na locahostu, pokud by byl problém a nic by se nám nezobrazilo, tak je možné, že náš lokální server poslouchá na portu a tedy je potřeba ho nastavit. Polymer používá importy HTML k načtení komponent. Importy HTML zajišťují správu závislostí a ujistí se, že všechny prvky a jejich závislosti jsou načteny před použitím. K vytvoření záložek neboli menu pro navigaci mezi různými pohledy jsme využili prvek <paper-tabs>, který vlastně pracuje obdobně jako <select>, ale je nastylován jako sada záložek.

```
<core-toolbar>

<paper-tabs id="tabs" selected="all" self-end>
<paper-tab name="all">all</paper-tab>
<paper-tab name="favorites">Oblibene</paper-tab>
</paper-tabs>

</core-toolbar>
```

Výpis 5: Přidání záložek

Dále jsme museli vyřešit přepínání mezi záložkami. Využili jsme prvku <paper-tabs>, který nám zavolá core-select událost. Nyní jsme otestovali funkčnost na localhostu. Když jsme měli otevřenou konzoli, tak jsme si všimli, že se nám vrací dvě core-select události pokaždé, když se přepneme mezi záložkami. Tedy jeden pro předchozí zvolenou záložku a jeden pro nově zvolenou záložku. Takže <paper-tabs> prvek zdědí toto chování od <core-selector>, který podporuje jak jeden tak více výběrů.

Dále jsme museli vytvořit vlastní prvek, který nám zobrazí příspěvek. Konečný vzhled zahrnuje profilový obrázek, jméno, tlačítko oblíbené a místo pro obsah. Toto zahrnuje práci s shadow DOM, který nám zajišťuje přidávání do lokálního DOM stromu uvnitř DOM prvku. Nyní bylo nutné pohrát si se stylováním nového obsahu. V polymeru je řada nových CSS selektorů se kterými se pracuje.

```
 polyfill -next-selector { content: '.card-header␣h2'; }
.card-header ::content h2 {
margin: 0;
font-size: 1.8rem;
font-weight: 300;
}
 polyfill -next-selector { content: '.card-header␣img'; }
.card-header ::content img {
```

```
width: 70px;
margin: 10px;
}
```

Výpis 6: Stylování prvků pomocí polymeru

Zde ::content h2 vybere jakýkoliv h2 tag, který je distribuován skrz kurzor. Může zde také nastat problém u prohlížečů, které nepodporují shadow DOM. Ještě jsme museli naimportovat nové prvky do indexu. Nyní za pomoci polymeru, přesněji <post-service> prvku budeme stahovat. Toto je vlastně jednoduchá API pro jednoduchou sociální síť. Tedy museli jsme navázat data mezi posts property a lokální property.

```
<div layout vertical center>

<template repeat="{{post in posts}}">
<post-card>
<img src="{{post.avatar}}" width="70" height="70">
<h2>{{post.username}}</h2>
<p>{{post.text}}</p>
</post-card>
</template>

</div>
```

Výpis 7: Vytváření instance pro každou položku

Pomocí repeat= post in posts docílíme toho, že se vytvoří nová instance pro každou položku v posts. V každé instanci se individuální vazby jako post.username naplní odpovídajícími hodnotami pro danou položku. Jako poslední jsme vytvořili vlastnost oblíbené a její metodu. V této metodě je událost, která vyvolá stav vlastnosti. V prvku favorite se aktualizuje DOM kdykoliv, když se změní vlastnost hodnoty.

4.5 Vytváření MongoDB

V první fázi jsme vytvářeli JSON objekty, kdy jsme museli navrhnout strukturu dané databáze pro spediční firmu. Zde jsme strávili chvíli času, než jsme si uvědomili, co a jak se vlastně vytváří a hlavně aby to dávalo smysl a bylo to realizovatelné. Vytvořili jsme JSON modely pro obchodní podmínky, platební podmínky a skladový registr, kde jsme museli vzít v úvahu jakého typu bude zboží, druh, složení, váha atd. Když jsme si sepsali JSON objekty a dostali z toho model, museli jsme připravit WebStorm k aplikaci modelu. Tedy nainstalovali jsme si MongoDB pomocí npm install mongodb. Poté jsme se museli připojit pomocí Mongo klienta k běžící MongoDB instanci tím, že jsme specifikovali MongoDB URI.

```
var url = 'mongodb://localhost:27017/test';
MongoClient.connect(url, function(err, db) {
assert.equal(null, err);
console.log("Connected correctly to server.");
db.close();
});
```

Výpis 8: Připojení k MongoDB

Úspěšné připojení nám vypsalo hlášku, že jsme se úspěšně připojili k serveru. Vložili jsme dokument do kolekce, která se jmenovala spedice, a poté jsme zavolali funkci tohoto vloženého dokumentu. Opět úspěšné vložení nám vyhodí hlášku, že byl vložen dokument do kolekce spedice. Nyní jsme se dotázali na všechny dokumenty v kolekci, abychom si mohli zobrazit údaje z databáze a dále s nimi pracovat.

```
var findforwarding = function(db, callback) {
var cursor =db.collection('spedice').find( );
cursor.each(function(err, doc) {
assert.equal(err, null);
if (doc != null) {
console.dir(doc);
} else {
callback();
}
});
};
```

Výpis 9: Funkce k nalezení dokumentů v kolekci

Tímto jsme docílili toho, že se dotážeme všech dokumentů v kolekci spedice. Obdobně jsme si vytvořili funkci k mazání a aktualizaci, zde jsme museli myslet na jednu věc a to, že po aktualizaci bude dokument obsahovat jenom pole, které jsou v náhradním dokumentu.

5 Závěr

Praxe v této firmě pro mě byla velkým přínosem. Ze začátku jsem se seznamoval se zázemím tak velké společnosti, jak to vlastně ve světě chodí a jak se řeší velké projekty pro mezinárodní společnosti. Osobně jsem se podílel na projektu, který slouží pro spediční společnosti. Naskladnění, vyskladnění a GPS zaměření pomocí Google maps API. Dále jsem vytvářel databázi v MongoDB, seznámil jsem se a vyzkoušel si nový projekt s názvem Polymer-Project. Hodně zajímavou částí pro mě byla konfigurace webového serveru na Amazonu a taky práce s operačním systémem Linux na distribuci Elementary OS, kde jsem se naučil práci s projekty a jejich zabezpečením. Dalším přínosem pro mně je práce s Git, kdy jsem se naučil pracovat s repozitáři a takhle i práci v kolektivu. Při spolupráci jsem se toho hodně naučil a hlavně seznámil s novými technologiemi používaných v dnešní době pro vytváření velkým webových aplikací. Ze školy jsem měl základy a o některých technologiích věděl, ale větší část pro mě byla novinou a proto to pro mě bylo těžší v prvních fázích. S tvorbou databází jsem měl základní zkušenosti ze školy. Taky s prací v Linuxu mám základní zkušenosti ze školy. Avšak práce s ostatními technologiemi pro mě byla novinou. Chyběly mi především znalosti, které se týkaly nových webových trendů a práce v týmu.

Ondřej Korbela

6 Reference

[1] *N*ode.js. [online]. [cit. 2015]. Dostupné z: https://nodejs.org/

[2] *P*olymer-Project. [online]. [cit. 2015]. Dostupné z: https://www.polymer-project.org/0.5/

[3] *M*ongoDB. [online]. [cit. 2015]. Dostupné z: https://www.mongodb.org/

[4] *H*TML5. [online]. [cit. 2015]. Dostupné z: http://www.html5.cz/

[5] *C*SS3. [online]. [cit. 2015]. Dostupné z: http://www.w3schools.com/css/

[6] *A*ngularJS. [online]. [cit. 2015]. Dostupné z: https://angularjs.org/

[7] *G*itHub. [online]. [cit. 2015]. Dostupné z: https://github.com/

[8] *M*ongoose. [online]. [cit. 2015]. Dostupné z: http://mongoosejs.com/

[9] *E*lementary OS. [online]. [cit. 2015]. Dostupné z: http://elementary.io/

[10] *N*erelační Databáze. [online]. [cit. 2015]. Dostupné z: http://www.zdrojak.cz/serialy/nerelacni-databaze/